PHP İle Paket Geliştirme

PHP için paket geliştirme ve yayınlama kılavuzu.

Özgür Adem Işıklı

PHP İle Paket Geliştirme

PHP için paket geliştirme ve yayınlama kılavuzu.

Özgür Adem Işıklı

Bu kitap şu adreste satılmaktadır http://leanpub.com/phpilepaketgelistirme

Bu versiyon şu tarihte yayımlandı 2015-01-21

İçindekiler

Önsöz

Bundan yaklaşık olarak 10 yıl önce ilk programımı yazdığımda yaşadığım mutluluğu tarif etmem imkansız. Hala yeni bir şeyler geliştirirken aynı heyecanı yaşıyor ve yaptığım işten keyif alıyorum.

Ancak yazılım hayatım boyunca bir noktadan sonra tekrar tekrar aynı işi yaptığımın farkına vardım. Bu sorunu aşmak için "Nasıl yazdığım kodları pratik bir şekilde yeniden kullanabilirim?" sorusunu kendime yönelttim. Her zaman olduğu gibi yalnız değildim ve benden daha önce de bu sorunun sorulduğunu öğrendim.

Bu e-kitapta birlikte bu sorunun çözümlerinden sadece biri olan paket tabanlı mimariyi inceleyeceğiz. Neden gerekli olduğu, ne gibi kolaylıklar sağladığı ve nereden başlanması gerektiği gibi konular tek tek irdelenecektir.

Bu e-kitabı iyi bir şekilde kavrayabilmek için daha önce PHP ile uygulama geliştirmiş olmanızı tavsiye ederim. E-kitap, PHP'yi bilmeyen geliştiriciler için uygun değildir.

Yazım süresince kararsız kaldığım bir diğer konu ise **Git** kullanımının e-kitap kapsamına dahil olup olmamasıdır. Uzun bir düşünme sürecinden sonra, asıl konudan çok sapacağımız için ***Git*** kullanımını kapsama dahil etmeme kararı aldım. Bu da e-kitap okuyucularının, temel ***Git*** kullanımını biliyor olmalarını gerektirmektedir.

Umarım benim yazarken aldığım keyfi siz de okurken alabilirsiniz ve sizin için faydalı bir döküman olur. Bu çalışma hakkındaki tüm fikir, öneri ve görüşlerinizi **i.ozguradem@gmail.com** e-posta adresine ya da **@iozguradem** Twitter hesabına bildirebilirsiniz.

Ocak 2015

Giriş

Yolculuğumuza başlamadan evvel, neden paket tabanlı çalışmaya ihtiyacımız olduğunu irdelememiz gerekiyor. Paket tabanlı mimariye olan zaruretimizi ne kadar iyi anlarsak, o kadar hızlı yol alabiliriz.

1. Aynı Şey, Tekrardan

5 yıllık bir geliştirici olduğunuzu varsayarak işe başlayalım. Bu süre içerisinde PHP ile sayısız proje geliştirmiş olmanız muhtemeldir. Her yaptığınız projenizde bir oturum açma/kapatma, kullanıcı kaydı ya da yetkilendirme sistemi bulunabilir. Ancak 5 yıl yazılım dünyası için oldukça uzun bir süredir. Belki ilk projenizde herhangi framework kullanmıyordunuz fakat bugüne gelene kadar en az bir tanesini aktif olarak kullanmış olabilirsiniz. Eğer yarın başlayacağınız yeni bir projede, hazır bir paket kullanmayacaksanız ve basit bir oturum açma işlemine için sıfırdan kod yazacaksanız, gerçekten ***maceraperest*** olmalısınız. Eğer bunun yerine elinizin altında oturum yönetimi için kullanılacak hazır bir kod varsa, bunu kullanmanız kadar güzel bir şey olamaz.

Sorunumuz işte tam da bu noktada ortaya çıkıyor. Aslında, framework'lerin de ortaya çıkmasının nedeni benzerdir; sürekli yaptığımız işlemlerin rutinleştirilmesi. **Dont Repeat Yourself** prensibi bize ***kendimizi tekrar etmememizi*** söyler. Geliştirdiğimiz bir kodu tekrar tekrar kullanamıyorsak, büyük ihtimalle bir yerde hata yapıyoruz demektir.

2. Aksan Farkı

Tekrar tekrar yazdığımız kodların kullanılabilmesi için; kodlarımızın kendi arasında aynı ***aksan*** ile konuşması gerektiği gerçeğinin altını çizmemiz gerekiyor. Aksan ile kastedilen; yazdığımız kodların kendi arasında bir standardının olmasıdır. Standartlara bağlı kalınmaksızın, kendimizi tekrar etmeden ve tüm topluluğun kullanabileceği kodlar geliştirmek imkansıza yakındır.

Eskiye nazaran standartlar açısından PHP'de epey yol alındığı söyleyebilirim. Ancak standartların takibinde, PHP geliştiricilerin aynı hassasiyette olduğunu söylemek güç. Ne yazık ki bu konuda kat etmemiz gereken çok yol var. Bir paket geliştirmek için PHP dünyasında kabul edilen bazı standartları biliyor olmanız kabul edilmektedir. Her ne kadar bazı framework'ler kendilerine has yazım standartları belirleseler de, genel kabul PHP Framework Interop Group[1] tarafından belirlenen standartlardır.

[1] http://www.php-fig.org

3. PHP Tarafında Güncel Durum

PHP dünyası artık Laravel ve Symfony2 gibi harika frameworklere sahip. Özellikle Laravel'in Symfony'den aldığı paketler daha bir dikkatle incelenmelidir. (Yanlış okumadınız. Laravel, Symfony2 ile bazı paketleri ortak kullanmaktadır.) Bunu ilk duyduğumda ufak çaplı bir şok yaşamıştım. İki framework arasında ortak paket kullanımının nasıl mümkün olabileceğini kavramam biraz zamanımı aldı. Fakat biz geliştiriciler birbirimize rakip değiliz (Ticari değeri olan ürünlerimiz elbette rakip ama ürünlerimizi geliştirirken kullandığımız araçlar benim bakış açımdan rakip değildir). Genelde birbirimizi rakip olarak görüp hazır bir kütüphane kullanmak yerine bir başka kütüphane geliştirmeyi tercih ediyoruz. Ancak bu çoğu zaman kaynak israfından öte bir katkı sağlamıyor. Biz e-kitap boyunca paket geliştirme işlemini anlatacak olsak da, sizin paket geliştirmeden önce aynı işi yapan paketleri bularak, projelerinize dahil etmenizi şiddetle ***öneriyoruz***. Lütfen var olan bir paketi yeniden yazmayınız.

Bilgi

Geçmişte ben de bir paket var olmasına rağmen kendim paket yazdım ve bunların çoğu hüsranla sonuçlandı. Hala GitHub hesabımda duran bazı paketleri de o an öğrendiğim konular üzerinde pratik kazanmak amacıyla geliştirdim. Tekrar altını çiziyorum; ***var olan paketi yeniden yazmayın***.

4. Kopyala/Yapıştır

Yönlendirme işlemleri her PHP projesinde, hatta her framework'de ortak olarak olması gereken bir özelliktir. Peki biz neden her projemizde aynı yönlendirme paketini kullanmayalım?

Son derece basit bir yönlendirme işlemini yapan, aşağıdaki kod satırı inceleyelim;

```
1  include($_GET['rota'].'.php');
```

Bundan 5 yıl önce amatör olarak geliştirdiğimiz bu yönlendirmeyi eğer bir paket olarak geliştirmiş olsaydık, her geliştirmeden sonra çok güçlü bir yönlendirme paketine sahip olabilirdik. Paketimiz güçlendikçe, her projemizde aynı paketi kullanabilirdik. Eğer yönlendirmede bir hata bulsaydık, tüm projelerimizde aynı hatayı düzeltebilirdik.

Bilgi

Tabi ki paket geliştirebilmeniz için, ***PHP***'nin bir paket/bağımlılık yöneticisine sahip olması gerekmektedir. Önceki yıllarda PHP dünyasında bu sorun varken, günümüzde **Composer** ile birlikte bir bağımlılık yöneticisine kavuşmuş bulunmaktayız. İlerleyen bölümlerde ***Composer***'a etraflıca değinilecektir.

5. Tek Kodla N Sayıda Proje

Örneğimizi daha iyi anlaşılabilmesi için değişik bir bakış açısı ile olaya bakmak istiyorum. İlk geliştirdiğimiz sitede bir yetkilendirme sistemi oluşturduğumuzu varsayalım. İkinci aldığımız işte yetkilendirme sistemini kopyala yapıştır yöntemiyle bir başka projemizde kullandığımızı varsayalım. Bu işlemi on farklı projenizde uyguladığınızı ve her seferinde paketi biraz daha geliştirdiğinizi düşünün. İlk yetkilendirme sisteminde 100, son sistemde 1000 satır kodunuz olsun. Lakin son sistemde ölümcül bir hata gördünüz ve mutlaka düzeltilmesi gerekiyor. 3 farklı dosyada, toplamda 45 satırı yeniden düzenlediniz. Bunu geçmişe dönük olarak projelerinize nasıl uygulayacaksınız? Eğer bu yöntemle değil de, paket yöntemiyle çalışmış olsaydık; bulduğumuz bir hatada yapacağımız düzeltmeyi çok kısa bir sürede tüm projelerimize aktarabilirdik.

6. Sonuç

Tüm bunlar anlattıklarımız bize sadece genel bir bakış açısı kazandırmaktadır. Ancak paket tabanlı çalışmak birçok zorunluluğu da beraberinde getirecektir. Standartlara ne kadar uyarsak, bir üst paragrafta bahsettiğim yöntemi o kadar başarılı bir şekilde uygulayabiliriz. Bundan sonra adım adım paket geliştirme süreçlerinin öncesini ve sonrasını irdeleyeceğiz.